ON SECURING CLOUD-HOSTED CYBER-PHYSICAL SYSTEMS USING TRUSTED EXECUTION ENVIRONMENTS

Amir Mohammad Naseri, Walter Lucia, Mohammad Mannan, Amr Youssef

Concordia Institute for Information Systems Engineering (CIISE)
Concordia University, Montreal, Canada

ABSTRACT

Recently, cloud control systems have gained increasing attention from the research community as a solution to implement networked cyber-physical systems (CPSs). Such an architecture can reduce deployment and maintenance costs albeit at the expense of additional security and privacy concerns. In this paper, first, we discuss state-of-the-art security solutions for cloud control systems and their limitations. Then, we propose a novel control architecture based on Trusted Execution Environments (TEE). We show that such an approach can potentially address major security and privacy issues for cloud-hosted control systems. Finally, we present an implementation setup based on Intel Software Guard Extensions (SGX), and validate its effectiveness on a testbed system.

Index Terms— Encrypted Control Systems, Trusted Execution Environments, Cloud/Edge-based CPS.

1. INTRODUCTION

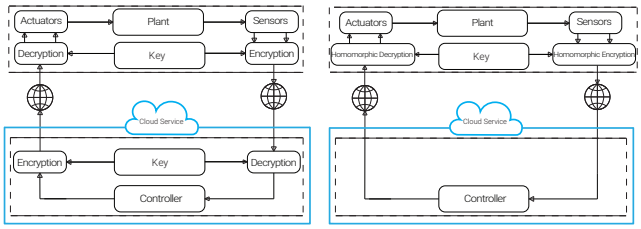
With the development of cloud services, the implementation of industrial control systems into the cloud/edge has received increasing attention. The use of such services saves on the cost of setting up and maintaining industrial control systems (ICSs), as well as off-loading computationally expensive tasks. Moreover, when ICSs are geographically distributed, these cloud services are highly available and accessible from different locations [1]. When using cloud services in such applications, the main concern is the security and privacy of the cloud environment and communication channels between the physical plant and the cloud-hosted controller.

Different approaches have been proposed to enhance the security and privacy of networked CPSs where the controller is hosted in a cloud infrastructure. For example, Zhou et al. [2] propose the use of conventional cryptographic algorithms to secure plant-to-cloud communication. Kogiso and Fujita [3] propose the use of homomorphic encryption to ensure that the controller's operations can be performed without decrypting the received data, and hence addressing the confidentiality problem in the cloud (in addition to securing communication channels). Homomorphic encryption-based solutions have received increasing attention by the CPSs community; for full homomorphic and Pailier's homomorphic based solutions, see e.g., [4–7]. However, these homomorphic solutions suffer from unavoidable limitations related to the arithmetic operations allowed by the homomorphic schemes, ciphertext size explosion, and computation overhead. For solutions targeting only securing communication channels cannot protect controller logic and data against a malicious or compromised cloud provider. For data and execution security in the context of IoT and CPS applications, Shepherd et al. [8] survey and compare several existing secure and trusted computing environments such as Trusted Platform Module (TPM), Secure Elements (SE), Trusted Execution Environments (TEEs), and Encrypted Execution Environments (E3).

In this paper, we explore the use of encryption and trusted execution environments to secure plant-to-cloud communication channels and protect data and controller logic for cloud-hosted/edge-hosted CPS applications. To understand the performance implications of our approach, we also design and implement a simple prototype for a quadruple tank system [9], using Intel SGX as our TEE. Our results indicate that the introduced overhead is negligible, and highly scalable yet secure CPS applications can be designed for a cloud/edge-deployment scenario. We hope that our initial results may be useful to the CPS security community and encourage the design of more efficient and secure TEE-based solutions compared to current schemes that rely mostly on conventional cryptographic mechanisms and homomorphic schemes.

2. SYSTEM SETUP AND THREAT MODEL

A typical cloud-based, networked control system consists of the following main components: the plant, the controller, the cloud, and the communication channels. The *plant* is the physical entity that we want to control. It is usually equipped with a set of *actuators* and *sensors*. The *controller* collects the sensor measurements and computes, according to a predefined control logic, the control commands sent to the actuators. In a cloud-based networked setup, the controller and the plant are spatially distributed, and the controller logic is implemented in a *cloud service* provider. The communication channels are used for a real-time and bi-directional ex-



(a) Encrypted communications. (b) Homomorphic encryption.

Fig. 1: Existing security solutions for cloud-based CPSs.

change of data (e.g., sensor measurements and control inputs) between the plant and the controller.

Threat Model. We consider the following attacks that can affect the privacy/security of the cloud-based CPS controllers.

Attacks against the communication channels - By adopting the conventional Dolev-Yao threat model [10], a malicious entity with access to the public communication channels is assumed to be able to eavesdrop on the transmitted data and/or modify their content. Therefore, potentially, the confidentiality and the integrity of the control system could be compromised. Indeed, such attackers can exploit the eavesdropped data to gain further information about the controlled system's behaviour and use their disruptive capabilities to launch sophisticated undetectable attacks such as replay, covert, zero-dynamics attacks [11, 12].

Attacks against the cloud service - If the cloud operator is malicious, or if the service is vulnerable, then an unauthorized entity (e.g., malware authors) might be able to gain access to the data transmitted between the plant and the controller, even if encrypted and authenticated communications are used. Indeed, such attackers could read the encryption key (key-management problem), intercept the transmitted data after decryption, and change the control logic (with the consequence of jeopardizing the whole control loop).

3. EXISTING SOLUTIONS

Different schemes have been proposed to secure networked control systems. A common solution is to use encrypted authenticated communications between the plant and the controller [13]; see Fig. 1a. Such a solution, at the cost of increased computational power to perform encryption/decryption operations at both the plant and controller's sides of the CPS, can mitigate the privacy and security issues related to cyber-attacks against the communication infrastructure. On the other hand, it does not address the security and privacy risks associated with the controller's deployment inside the cloud.

The use of homomorphic encryption has also been proposed to secure CPS solutions [3, 14]; see Fig 1b. A distinctive capability of such a solution is that it allows the controller to implement the control logic (in terms of additions and multiplications operations) directly on the received encrypted sensor measurements. Consequently, such an approach has the advantage of securing the communications while solving the privacy issues associated with the cloud infrastructure. However, common drawbacks of homomorphic encryption include: the mathematical operations performed on the encrypted data are typically limited and computationally expensive; and the plaintext to ciphertext bit expansion factor is usually very high. Consequently, homomorphic-based solutions might not be practical for securing industrial control systems with fast sampling rates or narrow bandwidth.

There are three different types of homomorphic encryption schemes, namely partially homomorphic encryption (PHE), somewhat homomorphic encryption (SHE), and fully homomorphic encryption (FHE). Each subclass is characterized by the set and number of encrypted operations allowed. Therefore, according to the limitations imposed by the used scheme, it might be challenging to recast any existing control algorithm into its encrypted counterpart. For example, FHE allows an unlimited number of encrypted addition and multiplication operations. Therefore it is particularly appealing to implement sophisticated control solutions such as dynamic feedback control or model predictive control. However, such freedom comes with a computational expensive bootstrapping process that makes FHE impractical to most control systems. Kim et al. [4] propose FHE to implement a dynamic output feedback controller using multiple controllers to avoid the bootstrapping delay. However, another inherent issue with FHE is that the ciphertext expansion might be up to $10000:1$ for an acceptable level of security of 100 bits [15]. Pailier's homomorphic encryption (PHE, supporting only encrypted additions) has also been proposed to implement a variety of controllers [5, 6]. However, due to memory issues related to the state of the dynamic encrypted controller (i.e., the number of bits required for its representation grows linearly with the number of iterations), the solution is usually limited to the use of resetting dynamics control laws. On the other hand, if a proportional controller is used, then, the control gain must satisfy some restrictive conditions imposed by the number of available bits [7].

Overall, existing solutions pose several limitations in terms of security/privacy/deployability to networked control systems. Moreover, no solutions have been proposed to protect CPSs against a malicious cloud operator, or malware that might be able to compromise the integrity of the control algorithm running on the cloud server.

4. OUR PROPOSAL

The objectives of our proposal are: secure the cloud-based CPSs against all the cyber-threat discussed in Section 2, and reduce the impact on the design and implementation of existing control strategies. The proposed secure control architecture has two essential components (see Fig. 2): an authenticated encryption scheme for securing the communication channels, and a TEE where the control logic is executed and the secret cryptographic keys, used by the authenticated encryption scheme, are stored.

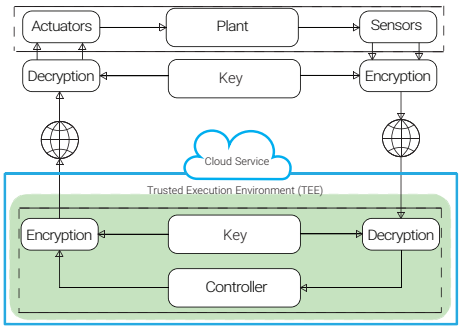


Fig. 2: Proposed TEE-based solution.

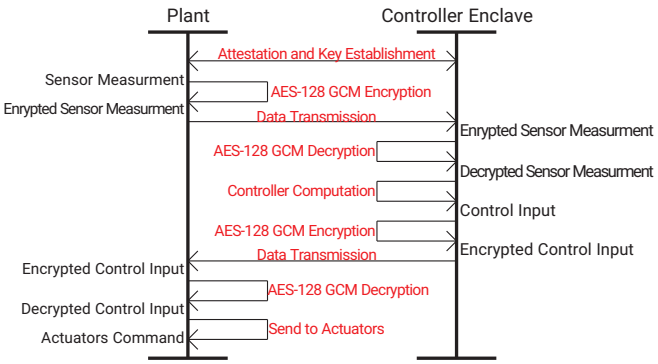


Fig. 3: Data flow in the proposed solution.

First, we resort to authenticated encryption schemes (cf. [13]) to ensure the integrity and the confidentiality of the control signal and sensor measurements exchanged between the plant and the controller. The used encryption scheme must be characterized by an inherent latency much smaller than the control-loop sampling time. The latter requirement is essential to ensure that the encryption scheme does not affect the control-loop system's stability. Second, a trusted execution environment (TEE) is used to protect the controller's operations in the cloud service. Generally speaking, a TEE refers to a hardware-based solution capable of ensuring that no malicious cloud entities (e.g., malware or a malicious cloud operator) could interfere with the execution of the control algorithm or with the memory associated with it. Moreover, if encryption/decryption operations are executed inside the TEE, where the keys are also protected by the TEE, then a malicious cloud administrator also cannot access the keys. TEE may also provide some other advantages such as measuring the integrity of the launched processes, measuring the origin of the TEE and current state of the TEE (attestability), and recovering the state of the TEE to a known good state after any corruption (recoverability). The presence of a TEE on the plant side is not required for our threat model. However, it is desirable in a scenario where the local computing platform (e.g., SCADA system) could be subject to cyber-attacks. Several solutions have been proposed in the literature (not in CPS) using different TEE implementations, e.g., Intel SGX [16], ARM TrustZone [17], AMD SEV [18], Hardware Security Module (HSM) [19], and secure co-processors [20]. Although all these solutions provide strong security mechanisms, not all can be used in our design (e.g., HSMs do not support remote attestation as opposed to Intel SGX).

5. IMPLEMENTATION

We use Intel SGX as TEE for its capability of providing a cryptographic attestation to ensure the integrity of the execution of the controller algorithm, even in the presence of a malicious cloud admin or a compromised cloud operating system (e.g., by a malware). To keep code and data secure, SGX provides an isolated execution environment, and encrypted memory. This secure container is called an "enclave" and everything else outside the enclave is assumed to be insecure. Two main functions are available to interact with the enclave, namely Enclave Call (E-CALL) and Out Call (O-CALL). E-CALL is used to call, from outside the enclave, a function implemented inside the enclave. On the other hand, O-CALL is used to call, from inside the enclave, a function implemented outside the enclave. For the implementation of the authenticated encryption, AES-128 Galois/Counter Mode(GCM) is used. This algorithm is a good candidate for CPSs because of its high throughput and low latency [21, 22]. First, we need to create an enclave and allocate memory for the Enclave Page Cache (EPC). The process starts with the attestation of both the enclave (validity of the CPU's SGX support) and the code (validity of the binary executed within the enclave as the controller logic). During the attestation, entities also establish a secure session key. After these initialization operations, data transmission will be started between the participating entities, encrypted under the session key. The data flow for a single control loop is shown in Fig. 3. In particular, the sensor measurements are encrypted on the plant side. Then, these encrypted sensor measurements are sent to the cloud over the communication channel. The authenticity of the received measurement is checked inside the enclave, where then the controller logic is also applied to the decrypted measurements. The evaluated controller output is then encrypted (inside the enclave) before it is sent to the actuator through the communication channel. Finally, the encrypted control input is decrypted by the actuator and applied to the plant.

6. SECURITY AND PERFORMANCE EVALUATION

We now discuss the security properties of the proposed solution. (i) *Confidentiality:* Data sent through the communication channels are encrypted with AES-128 GCM. Therefore, network eavesdroppers are unable to decrypt the transmitted control signals and sensor measurements. Moreover, control operations and encryption/decryption operations are performed within the enclave, avoiding the possibility that a malware or cloud administrator could intercept the plaintext signals or acquire the keys. (ii) *Integrity:* By exploiting the

message authentication code (MAC) tag in AES-128 GCM, it is possible to verify the integrity of the transmitted data (i.e., detect if an attacker has manipulated the transmitted data). Another aspect of integrity is to make sure that the controller logic is not manipulated by the cloud provider before the code is executed within the enclave. For this purpose, an attestation operation is performed to make sure that the code executed in the enclave is exactly that is sent to the cloud service by the system admin. To improve code obfuscation (i.e., hiding the control logic from the cloud operator), the proposed solution in [23] can be used. Note that the controller's runtime state remains always protected by SGX's memory encryption. Moreover, since the controller is executed inside SGX, the integrity of the control algorithm is also ensured. (iii) *Authentication:* The remote attestation feature of Intel SGX is used on the plant side to establish a secure and authenticated communication channel with the enclave in the cloud and ensure that the remote enclave is trusted. The MAC tags also is used by both entities (plant and controller) to make sure that the received messages are obtained by a trusted entity. (iv) *Freshness:* The uniqueness of the AES-128 GCM IV is used to guarantee freshness of each message. Defending against side-channel attacks against Intel SGX [24] is outside the scope of this paper. In the case of necessity of storing data by the controller (depend on the controller logic), to mitigate rollback attacks on the sealed data, Monotonic Counter (MC) of Intel SGX can be used to guarantee that the sealed data is the latest copy.

6.1. Performance Evaluation

System setup. As a testbed, we use the Quadruple Tank Process (QTP) system from Johansson [9], which is often used as a benchmark for control systems applications. The system consists of four water tanks where $h_i, i \in 1,2,3,4$ represents the level of water in each tank and also represents the states x of the system, i.e., $x = [h_1,\, h_2,\, h_3,\, h_4]^2 \in \mathbb{R}^4$. There are two sensors that measure the level of water inside tanks 1 and 2, i.e., the output measurement vector is $y = [0.5h_1,\, 0.5h_2]^T \in \mathbb{R}^2$. Moreover, the system is equipped with two pumps and the applied voltage v_1, v_2 are the inputs u of the system, i.e., $u = [v_1,\, v_2]^T$. We have linearized the system model around the equilibrium pair ($x_{eq} = [12.4,\, 12.7\,, 1.8,\, 1.4]^T$ $u_{eq} = [3,\, 3]^T$) and discretized it using a sampling time $T_s = 0.1\,\text{sec}$. The linearized model $x(k+1) = Ax(k) + Bu(k)$, $y(k) = Cx(k)$ and its matrices A, B, C can be easily obtained following [9]. The plant is regulated by means of dynamic output feedback controller consisting of a Luenberger Observer and an optimal Linear Quadratic (LQ) controller. The state-estimator operations are described by the discrete-time system $\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k))$ where $\hat{x}(k)$ is the estimation of the state $x(k)$ and the correction gain is given by $L = \begin{bmatrix} 0.78 & 0 & 0.32 & 0 \\ 0 & 0.78 & 0 & 0.32 \end{bmatrix}^T$. The LQ controller logic is computed as $u = K(x - x_{eq}) + u_{eq}$ where the stabilizing gain is given by $K = \begin{bmatrix} 27.547 & -0.054 & 0.468 & 0.086 \\ 0.023 & 28.441 & 0.143 & 0.507 \end{bmatrix}$. The dynamic output feedback controller operations have been implemented by utilizing an Intel SGX running on an Intel Core i7-6700 CPU, 3.40GHz, with 4 cores and 8 threads and 16 GB of RAM, using 64-bit Windows 7.

Measurements. We have conducted a series of measurements to evaluate the computation times required by different components of the proposed solution (see the data flow in Fig. 3). The reported CPU measurements have been obtained using the approach proposed in [25, Fig. 1], i.e., an O-CALL function is used as a stopwatch. As a result, the time measurements in Table 1 include an extra time representing the CPU time required to return to the enclave from an O-CALL and exit from it. We denote this time by Δt. To mitigate the presence of Δt in the measurements, we repeated each operation inside the enclave 1000 times and then calculate the average. Δt is also measured separately. The numerical results show that the two dominant factors are Δt and the control algorithm CPU time. Indeed, the average total CPU time required by both the secure and insecure implementations are around $905\mu s$ and $479\mu s$, respectively. The obtained results confirm that the computational overhead introduced by the use of Intel SGX does not affect the feasibility of the control strategy. Moreover, given that the introduced overhead is in the milliseconds' range, the proposed SGX-based secure architecture is believed to be affordable for a large class of cloud-based control systems applications.

Operation	Time (μs)
Enclave creation	8368.4
Dynamic output feedback controller	466.7
AES-128 GCM encryption	1.8
AES-128 GCM decryption	1.4
Δt	435.4

Table 1: Average time for different operations of the SGX-based solution

7. CONCLUSION

We proposed a solution to secure cloud-hosted/edge-hosted CPSs. In particular by resorting to authenticated encryption and a trusted execution environment, we showed that the proposed networked control scheme is secure again different attacks against its security and privacy. We verified the effectiveness of such a scheme by means of numerical simulations obtained considering Intel SGX, where we performed different benchmarks to evaluate the computational burden associated to the trusted control scheme implementation. The obtained results show good promise in terms of real-time performance and simplicity of implementation in CPSs applications. The proposed solution can also be implemented in a non cloud setting to help mitigating supply chain breaches.

8. REFERENCES

[1] M. S. Mahmoud and Y. Xia, *Networked control systems: cloud control and secure control*, Butterworth-Heinemann, 2019.

[2] L. Zhou, V. Varadharajan, and M. Hitchens, "Achieving secure role-based access control on encrypted data in cloud storage," *IEEE trans. on information forensics and security*, vol. 8, no. 12, pp. 1947–1960, 2013.

[3] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *IEEE Conf. on Decision and Control (CDC)*. IEEE, 2015, pp. 6836–6843.

[4] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[5] J. Tran, M. Farokhi, F.and Cantoni, and I. Shames, "Implementing homomorphic encryption based secure feedback control," *Control Engineering Practice*, vol. 97, pp. 104350, 2020.

[6] C. Murguia, F. Farokhi, and I. Shames, "Secure and private implementation of dynamic controllers using semi-homomorphic encryption," *IEEE Trans. on Automatic Control*, vol. 65, no. 9, pp. 3950–3957, 2020.

[7] Y. Lin, F. Farokhi, I. Shames, and D. Nešić, "Secure control of nonlinear systems using semi-homomorphic encryption," in *IEEE Conf. on Decision and Control*, 2018, pp. 5002–5007.

[8] C. Shepherd, G. Arfaoui, I. Gurulian, R. P. Lee, K. Markantonakis, D. Akram, R. N.and Sauveron, and E. Conchon, "Secure and trusted execution: Past, present, and future-a critical review in the context of the internet of things and cyber-physical systems," in *IEEE Trustcom/BigDataSE/ISPA*, 2016, pp. 168–177.

[9] K. H. Johansson, "The quadruple-tank process: A multivariable laboratory process with an adjustable zero," *IEEE Trans. on control systems Tech*, vol. 8, no. 3, pp. 456–465, 2000.

[10] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[11] S.M. Dibaji, M. Pirani, D. B. Flamholz, A. M. Annaswamy, K. H. Johansson, and A. Chakrabortty, "A systems and control perspective of CPS security," *Annual Reviews in Control*, 2019.

[12] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "A secure control framework for resource-limited adversaries," *Automatica*, vol. 51, pp. 135–148, 2015.

[13] S.C. Patel, G. D. Bhatt, and J. H. Graham, "Improving the cyber security of SCADA communication networks," *Communications of the ACM*, vol. 52, no. 7, pp. 139–142, 2009.

[14] F. Farokhi, I. Shames, and N. Batterham, "Secure and private cloud-based control using semi-homomorphic encryption," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 163–168, 2016.

[15] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "Tfhe: fast fully homomorphic encryption over the torus," *Journal of Cryptology*, vol. 33, no. 1, pp. 34–91, 2020.

[16] V. Costan and S Devadas, "Intel SGX explained.," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

[17] AMBA Infrastructure, "Technical overview," 2004.

[18] D. Kaplan, J. Powell, and T. Woller, "Amd memory encryption," *White paper*, 2016.

[19] J. Varia, S. Mathew, and et. al, "Overview of amazon web services," *Amazon Web Services*, vol. 105, 2014.

[20] S. Bajaj and R. Sion, "Trusteddb: A trusted hardware-based database with privacy and data confidentiality," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 752–765, 2013.

[21] Sandhya Koteshwara, Amitabh Das, and Keshab K Parhi, "FPGA implementation and comparison of AES-GCM and Deoxys authenticated encryption schemes," in *IEEE Int. symposium on circuits and systems*, 2017, pp. 1–4.

[22] V. Arun, K. Vanisree, and D. Reddy, "Implementation of AES-GCM encryption algorithm for high performance and low power architecture using FPGA," 2015.

[23] E Bauman, H Wang, M Zhang, and Z Lin, "Sgxelide: enabling enclave code secrecy via self-modification," in *Proceedings of Int. Symposium on Code Generation and Optimization*, 2018, pp. 75–86.

[24] F. Brasser, U. Müller, A. Dmitrienko, S. Kostiainen, K.and Capkun, and A.-R. Sadeghi, "Software grand exposure: SGX cache attacks are practical," in *USENIX Workshop on Offensive Tech.*, 2017.

[25] A. T. Gjerdrum, R. Pettersen, H. D. Johansen, and D. Johansen, "Performance principles for trusted computing with intel SGX," in *Int. Conf. on Cloud Computing and Services Science*. Springer, 2017, pp. 1–18.